# APPLICATION FOR UNITED STATES LETTERS PATENT

APPLICANTS: WILLIAM JOHN GOETZINGER
GLEN HOWARD HANDLOGTEN
JAMES FRANCIS MIKOS
DAVID ALAN NORGAARD
DANIEL JAMES SUCHER

TITLE: EMPTY INDICATORS FOR WEIGHTED FAIR QUEUES

DOCKET NO.: ROC920010202US1

**INTERNATIONAL BUSINESS MACHINES CORPORATION**

CERTIFICATE OF MAILING UNDER 37 CFR 1.10

I hereby certify that, on the date shown below, this correspondence is being deposited with the United States Postal Service in an envelope addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231, as "Express Mail Post Office to Addressee" Mailing Label No.: ET017005585US
on November 1, 2001

Name of person mailing papers: Brian M. Dugan

Signature Date 11/1/01

# EMPTY INDICATORS FOR WEIGHTED FAIR QUEUES

## CROSS REFERENCE TO RELATED PATENT APPLICATIONS

The present application is related to the following U.S. Patent Applications, each of which is hereby incorporated by reference herein in its entirety:

U.S. Patent Application Serial No. _____, filed _____, titled "WEIGHTED FAIR QUEUE HAVING EXTENDED EFFECTIVE RANGE" (IBM Docket No. ROC920010199US1);

U.S. Patent Application Serial No. _____, filed _____, titled "WEIGHTED FAIR QUEUE SERVING PLURAL OUTPUT PORTS" (IBM Docket No. ROC920010200US1);

U.S. Patent Application Serial No. _____, filed _____, titled "WEIGHTED FAIR QUEUE HAVING ADJUSTABLE SCALING FACTOR" (IBM Docket No. ROC920010201US1);

U.S. Patent Application Serial No. _____, filed _____, titled "QoS SCHEDULER AND METHOD FOR IMPLEMENTING PEAK SERVICE DISTANCE USING NEXT PEAK SERVICE TIME VIOLATED INDICATION" (IBM Docket No. ROC920010203US1);

U.S. Patent Application Serial No. _____, filed _____, titled "QoS SCHEDULER AND METHOD FOR IMPLEMENTING QUALITY OF SERVICE WITH AGING STAMPS" (IBM Docket No. ROC920010204US1);

U.S. Patent Application Serial No. _____, filed _____, titled "QoS SCHEDULER AND METHOD FOR IMPLEMENTING QUALITY OF SERVICE WITH CACHED STATUS ARRAY" (IBM Docket No. ROC920010205US1); and

U.S. Patent Application Serial No. _____, filed _____, titled "QoS SCHEDULER AND METHOD FOR IMPLEMENTING QUALITY OF SERVICE ANTICIPATING THE END OF A CHAIN OF FLOWS" (IBM Docket No. ROC920010206US1).

## FIELD OF THE INVENTION

The present invention is concerned with data and storage communication systems and is more particularly concerned with a scheduler component of a network processor.

## BACKGROUND OF THE INVENTION

Data and storage communication networks are in widespread use. In many data and storage communication networks, data packet switching is employed to route data packets or frames from point to point between source and destination, and network processors are employed to handle transmission of data into and out of data switches.

FIG. 1 is a block diagram illustration of a conventional network processor in which the present invention may be applied. The network processor, which is generally indicated by reference numeral 10, may be constituted by a number of components mounted on a card or "blade". Within a data communication network, a considerable number of blades containing network processors may be interposed between a data switch and a data network.

The network processor 10 includes data flow chips 12 and 14. The first data flow chip 12 is connected to a data switch 15 (shown in phantom) via first switch ports 16, and is connected to a data network 17 (shown in phantom) via first network ports 18. The first data flow chip 12 is

positioned on the ingress side of the switch 15 and handles data frames that are inbound to the switch 15.

The second data flow chip 14 is connected to the switch 15 via second switch ports 20 and is connected to the data network 17 via second network ports 22. The second data flow chip 14 is positioned on the egress side of the switch 15 and handles data frames that are outbound from the switch 15.

As shown in FIG. 1, a first data buffer 24 is coupled to the first data flow chip 12. The first data buffer 24 stores inbound data frames pending transmission of the inbound data frames to the switch 15. A second data buffer 26 is coupled to the second data flow chip 14, and stores outbound data frames pending transmission of the outbound data frames to the data network 17.

The network processor 10 also includes a first processor chip 28 coupled to the first data flow chip 12. The first processor chip 28 supervises operation of the first data flow chip 12 and may include multiple processors. A second processor chip 30 is coupled to the second data flow chip 14, supervises operation of the second data flow chip 14 and may include multiple processors.

A control signal path 32 couples an output terminal of second data flow chip 14 to an input terminal of first data flow chip 12 (e.g., to allow transmission of data frames therebetween).

The network processor 10 further includes a first scheduler chip 34 coupled to the first data flow chip 12. The first scheduler chip 34 manages the sequence in which inbound data frames are transmitted to the switch 15 via first switch ports 16. A first memory 36 such as a fast

SRAM is coupled to the first scheduler chip 34 (e.g., for storing data frame pointers and flow control information as described further below). The first memory 36 may be, for example, a QDR (quad data rate) SRAM.

A second scheduler chip 38 is coupled to the second data flow chip 14. The second scheduler chip 38 manages the sequence in which data frames are output from the second network ports 22 of the second data flow chip 14. Coupled to the second scheduler chip 38 are at least one and possibly two memories (e.g., fast SRAMs 40) for storing data frame pointers and flow control information. The memories 40 may, like the first memory 36, be QDRs. The additional memory 40 on the egress side of the network processor 10 may be needed because of a larger number of flows output through the second network ports 22 than through the first switch ports 16.

FIG. 2 schematically illustrates conventional queuing arrangements that may be provided for a data flow chip/scheduler pair (either the first data flow chip 12 and the first scheduler chip 34 or the second data flow chip 14 and the second scheduler chip 38) of the network processor 10 of FIG. 1. In the particular example illustrated in FIG. 2, the first data flow chip 12 and the first scheduler chip 34 are illustrated, but a very similar queuing arrangement may be provided in connection with the second data flow chip 14 and the second scheduler chip 38. In the queuing arrangement for the first data flow chip 12 and the first scheduler chip 34, incoming data frames (from data network 17) are buffered in the input data buffer 24 associated with the first data flow chip 12 (FIG. 1). Each data frame is associated with a data flow or "flow". As is familiar to

those who are skilled in the art, a "flow" represents a one-way connection between a source and a destination.

Flows with which the incoming data frames are associated are enqueued in a scheduling queue 42 maintained in the first scheduler chip 34. The scheduling queue 42 defines a sequence in which the flows enqueued therein are to be serviced. The particular scheduling queue 42 of interest in connection with the present invention is a weighted fair queue which arbitrates among flows entitled to a "best effort" or "available bandwidth" Quality of Service (QoS).

As shown in FIG. 2, the scheduling queue 42 is associated with a respective output port 44 of the first data flow chip 12. It is to be understood that the output port 44 is one of the first switch ports 16 illustrated in FIG. 1. (However, if the data flow chip/scheduler pair under discussion were the egress side data flow chip 14 and scheduler chip 38, then the output port 44 would be one of the network ports 22.) Although only one scheduling queue 42 and one corresponding output port 44 are shown, it should be understood that in fact there may be plural output ports and corresponding scheduling queues each assigned to a respective port. (However, according to an alternative embodiment, disclosed in co-pending patent application Serial No. _____, filed _____ (Attorney Docket No. ROC920010200US1), a group of output ports may be associated with each scheduling queue 42. This co-pending patent application is incorporated herein by reference.)

Although not indicated in FIG. 2, the first scheduler chip 34 also includes flow scheduling calendars which define output schedules for flows which are entitled

to a scheduled QoS with guaranteed bandwidth, thus enjoying higher priority than the flows governed by the scheduling queue 42.

The memory 36 associated with the first scheduler chip 34 holds pointers ("frame pointers") to locations in the first data buffer 24 corresponding to data frames associated with the flows enqueued in the scheduling queue 42. The memory 36 also stores flow control information, such as information indicative of the QoS to which flows are entitled.

When the scheduling queue 42 indicates that a particular flow enqueued therein is the next to be serviced, reference is made to the frame pointer in the memory 36 corresponding to the first pending data frame for the flow in question and the corresponding frame data is transferred from the first data buffer 24 to an output queue 46 associated with the output port 44.

A more detailed representation of the scheduling queue 42 is shown in FIG. 3. As noted above, the scheduling queue 42 is used for weighted fair queuing of flows serviced on a "best effort" basis. In a particular example of a scheduling queue as illustrated in FIG. 3, the scheduling queue 42 has 512 slots (each slot represented by reference numeral 48). Other numbers of slots may be employed. In accordance with conventional practice, flows are enqueued or attached to the scheduling queue 42 based on a formula that takes into account both a length of a data frame associated with a flow to be enqueued and a weight which corresponds to a QoS to which the flow is entitled.

More specifically, the queue slot in which a flow is placed upon enqueuing is calculated according to the

formula CP + ((WF x FS)/SF), where CP is a pointer ("current pointer") that indicates a current position (the slot currently being serviced) in the scheduling queue 42; WF is a weighting factor associated with the flow to be enqueued, the weighting factor having been determined on the basis of the QoS to which the flow is entitled; FS is the size of the current frame associated with the flow to be enqueued; and SF is a scaling factor chosen to scale the product (WF x FS) so that the resulting quotient falls within the range defined by the scheduling queue 42. (In accordance with conventional practice, the scaling factor SF is conveniently defined as a integral power of 2 -- i.e., $SF = 2^n$, with n being a positive integer -- so that scaling the product (WF x FS) is performed by right shifting.) With this known weighted fair queuing technique, the weighting factors assigned to the various flows in accordance with the QoS assigned to each flow govern how close to the current pointer of the queue each flow is enqueued. In addition, flows which exhibit larger frame sizes are enqueued farther from the current pointer of the queue, to prevent such flows from appropriating an undue proportion of the available bandwidth of the queue. Upon enqueuement, data that identifies a flow (the "Flow ID") is stored in the appropriate queue slot 48.

As noted above, each scheduler may include a plurality of scheduling queues. For example, 64 scheduling queues may be supported in each scheduler. Each scheduling queue services a respective output port, or a group of output ports as taught in the above-referenced co-pending patent application Serial No. _____.

The scheduling queues may be accessed one after another in accordance with a round robin process, to search the scheduling queues for respective flows to be dequeued. One scheduling queue may be searched during each operating cycle of the scheduler. However, if the scheduling queue that is searched during a given cycle turns out to be empty, then the cycle may be wasted.

It is known to provide a counter for each scheduling queue to keep track of whether or not the scheduling queue is empty. However, operation of each counter may entail two increment operations (one for attachment of a new flow, and one for reattachment of a previously attached flow) and one decrement operation (reflecting detachment of a winning flow) during each cycle. Thus using a counter to track whether or not a scheduling queue is empty may adversely affect the performance of the scheduler. Moreover, providing a counter for each queue adds to the complexity and space requirements of the scheduler design.

An improved technique for determining whether or not a scheduling queue is empty would therefore be desirable.

## SUMMARY OF THE INVENTION

According to an aspect of the invention, a scheduler for a network processor is provided. The scheduler includes one or more scheduling queues, each scheduling queue adapted to define a respective sequence in which flows are to be serviced. The scheduler further includes one or more empty indicators, with each empty indicator being associated with a respective scheduling

queue to indicate whether the respective scheduling queue is empty. Each empty indicator may be a bit in a register.

According to another aspect of the invention, a method of dequeuing a flow from a scheduling queue is provided. The method includes examining an empty indicator associated with the scheduling queue, and refraining from searching the scheduling queue if the empty indicator indicates that the scheduling queue is empty. The method further includes searching the scheduling queue if the empty indicator indicates that the scheduling queue is not empty, and detaching from the scheduling queue a winning flow found in the searching step. The examining step may include checking a bit in a register.

According to still another aspect of the invention, a method of enqueuing a flow to a scheduling queue includes attaching a flow to the scheduling queue, and placing an empty indicator associated with the scheduling queue in a condition to indicate that the scheduling queue is not empty. The placing step may include setting or resetting a bit in a register.

According to still a further aspect of the invention, a method of dequeuing a flow from a scheduling queue is provided. The method includes examining an empty indicator associated with the scheduling queue, and refraining from searching the scheduling queue if the empty indicator indicates that the scheduling queue is empty. The method further includes searching the scheduling queue if the empty indicator indicates that the scheduling queue is not empty. According to a further step of the method, if a winning flow is found in the searching step, the winning flow is detached from the scheduling queue. According to

still a further step, if no flow is found in the searching step, the empty indicator is placed in a condition to indicate that the scheduling queue is empty. The examining step may include checking a bit in a register.

With the present invention, the empty status of scheduling queues is tracked while minimizing the expenditure of processing and hardware resources.

Other objects, features and advantages of the present invention will become more fully apparent from the following detailed description of exemplary embodiments, the appended claims and the accompanying drawings.

## BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of a conventional network processor in which the present invention may be applied;

FIG. 2 is a block diagram representation of conventional queuing arrangements provided in a data flow chip/scheduler pair included in the network processor of FIG. 1;

FIG. 3 is a pictorial representation of a weighted fair queuing scheduling queue provided in accordance with conventional practices;

FIG. 4 is a schematic illustration of a manner in which empty indicators are examined and scheduling queues are searched in accordance with a round robin process provided in accordance with the invention;

FIG. 5 is a flow chart that illustrates a process for detecting empty scheduling queues and detaching flows from scheduling queues in accordance with the invention; and

FIG. 6 is a flow chart that illustrates an alternative process provided in accordance with the invention for detecting empty scheduling queues and detaching flows from scheduling queues.

DETAILED DESCRIPTION

Exemplary embodiments of the invention will now be described with reference to FIGS. 4-6.

FIG. 4 is a schematic illustration of a round robin process, provided in accordance with the invention, whereby empty indicators are examined and scheduling queues (also referred to as "rings") of the scheduler 49 are searched for winning flows. It is to be understood that a "winning flow" is a flow that is closest to the current pointer of a scheduling queue. The scheduler 49 may be similar to the conventional scheduler 34 of FIG. 1, but with the additional inventive features described below. In FIG. 4, reference numeral 50 indicates empty indicators, each of which is associated with a respective scheduling queue. For convenience only one scheduling queue (shown as "RING 2") is illustrated in FIG. 4. In one embodiment, each empty indicator 50 is constituted by a respective bit in a register (not separately shown). In general, any conventional register such as a shift register latch (SRL) or other storage means may be employed for each empty indicator 50. As used herein and in the appended claims "empty indicator" does not include a counter.

It will be appreciated that the drawing of FIG. 4 only shows a relatively small portion of a larger number of empty indicators 50 that may be included in a scheduler in accordance with the invention. For example, if 64

scheduling queues are employed within a scheduler, there will be 64 empty indicators each corresponding to a respective one of the 64 scheduling queues maintained in the scheduler. As stated, only one scheduling queue 42 is shown in FIG. 4 for the sake of simplicity. Furthermore, although a relatively large number of slots, such as 512 slots, may be included in each scheduling queue 42, to simplify FIG. 4 the scheduling queue 42 shown in FIG. 4 is illustrated with only about 20 slots. The slots are represented by filled-in circles (e.g., 48a), indicating that the respective slot is occupied by an entry for a flow attached to the scheduling queue 42, and by open circles (e.g., 48b), indicating that there is no entry in the respective slot and that the slot accordingly is empty. It will be observed that the scheduling queue 42 shown in FIG. 4 (indicated as being scheduling queue or ring number 2 of 64 scheduling queues) has a number of filled slots so that the scheduling queue 42 is not empty. Consequently, the corresponding empty indicator for the scheduling queue/ring 2 (represented by reference numeral 50-2) is shown as being filled in, to indicate that the scheduling queue/ring 2 is not empty. On the other hand, other empty indicators, such as 50-3 (which corresponds to another scheduling queue 42 (not shown) represented as "Ring 3"), are open ovals, to indicate that the corresponding scheduling queues (not shown) are empty.

During initialization of the scheduler 49, all of the empty indicators 50 may be initially placed in a condition to indicate that the respective scheduling queues 42 are empty. Thenceforward, each time a flow is attached or reattached to a scheduling queue 42, the corresponding

empty indicator 50 is forced to a condition which indicates that the corresponding scheduling queue 42 is not empty.

Placement of the empty indicators 50 into a condition which indicates that the respective scheduling queue 42 is empty, and dequeuing of flows from the scheduling queues 42, will now be described, initially with reference to FIG. 5. FIG. 5 begins with a block 60. In block 60, the scheduler 49 proceeds with a round robin process in which the scheduling queues 42 are traversed in sequence and are given respective opportunities to be serviced. Specifically, the sequence of scheduling queues 42 is traversed to find the next "active" scheduling queue 42.

A scheduling queue 42 is considered to be "active" if it is not empty, and if at least one output port assigned to the scheduling queue 42 is not in a backpressure condition. (The concept of backpressure is well known to those who are skilled in the art, and need not be explained herein.) Thus, for the scheduling queue 42 which follows the most recently searched scheduling queue 42, the corresponding empty indicator 50 is examined to determine whether the empty indicator 50 indicates that the associated scheduling queue 42 is empty. If the empty indicator 50 indicates that the scheduling queue 42 is empty, then the scheduling queue 42 is not searched, and the empty indicator 50 of the following scheduling queue 42 is examined. However, if the empty indicator 50 indicates that the scheduling queue 42 is not empty (and assuming that at least one output port assigned to the scheduling queue 42 is not in a backpressure condition), then the scheduling queue 42 is selected for searching. Searching of the scheduling

queue (ring) 42 is indicated at block 62 in FIG. 5. Searching of the scheduling queue 42 for a "winning" flow (i.e., the flow that is closest to the head of the scheduling queue) may proceed in a conventional fashion. Alternatively, the scheduling queue 42 may be constituted in accordance with an invention disclosed in co-pending patent application Serial No. _____, filed _____ (Attorney Docket No. ROC920010199US1), such that the scheduling queue has plural subqueues with mutually different ranges and resolutions. In this case, the plural subqueues may be searched in parallel to find the flow that is closest to the head of the scheduling queue. The disclosure of this co-pending patent application is incorporated herein by reference.

Following block 62 is decision block 64. In decision block 64, it is determined whether the search of the scheduling queue 42 (selected in block 60 and searched in block 62) has indicated that the scheduling queue 42 is empty. If so, then block 66 follows decision block 64. At block 66, the empty indicator 50 associated with the scheduling queue 42 is placed in a condition to indicate that the scheduling queue 42 is empty. This may be done, for example, by setting or resetting an appropriate bit in a register. Following block 66, the procedure of FIG. 5 loops back to block 60.

If at decision block 64 it is determined that the scheduling queue 42 (selected in block 60 and searched in block 62) was not found to be empty, then decision block 68 follows decision block 64. At decision block 68 it is determined whether a flow that is entitled to scheduled service, or another higher priority flow, is to be serviced

from the output port corresponding to the winning flow found at block 62. In other words, it is determined whether a higher priority flow preempts servicing of the winning flow from the scheduling queue 42 searched at block 62. If such is not the case, then block 70 follows decision block 68. At block 70 the winning flow from the scheduling queue 42 is detached from the scheduling queue and serviced in accordance with conventional practice. The procedure of FIG. 5 then loops back to block 60.

However, if at decision block 68 it is found that the winning flow from the scheduling queue 42 searched at block 62 is to lose out to a higher priority flow, then the procedure of FIG. 5 loops back to block 60 from decision block 68 without detaching the winning flow from the scheduling queue 42.

In accordance with the procedure of FIG. 5, when a scheduling queue 42 is searched and found to be empty, the empty indicator 50 is placed in a condition to indicate that the scheduling queue 42 is empty. Consequently, the next time the scheduling queue 42 is reached in the selection process (e.g., a round robin process), it will be possible to determine that the scheduling queue 42 is empty by reference to the empty indicator 50 and without searching the empty scheduling queue 42. As a result, the operation cycle of the scheduler 49 may be used for searching a scheduling queue 42 which is not known to be empty. (Of course, this assumes that no flow is attached to the scheduling queue 42 between the time the scheduling queue 42 is found to be empty and the next time that it is reached in the selection process. If a flow is attached to the scheduling queue 42 in the interim, then the empty indicator

50 associated with the scheduling queue 42 is placed in a condition to indicate that the scheduling queue 42 is not empty.)

An alternative procedure is provided in accordance with another aspect of the invention for situations in which a flow detached from a scheduling queue is the only flow that was enqueued in the scheduling queue. This alternative procedure is illustrated by the flow chart of FIG. 6. It will be observed that the flow chart of FIG. 6 has the same blocks 60-70 as the flow chart of FIG. 5. Blocks 60-70 perform the same functions in the procedure of FIG. 6 as in the procedure of FIG. 5 and accordingly need not be explained again. However, in the procedure of FIG. 6, block 70 (detachment of the winning flow from the scheduling queue 42) is followed by a block 72. At block 72, the winning flow is masked and a further search of the scheduling queue 42 (selected in block 60 and searched in block 62) is carried out to determined whether the scheduling queue 42 was empty but for the winning flow found at block 62 and detached at block 70.

Following block 72 is a decision block 74. If it is determined at decision block 74 that the scheduling queue 42 was empty but for the flow that was just detached, then block 66 follows block 74. As noted before in conjunction with FIG. 5, block 66 involves placing the corresponding empty indicator 50 for the scheduling queue 42 in a condition to indicate that the scheduling queue 42 is empty. Following block 66 the procedure of FIG. 6 loops back to block 60.

However, if it is determined at decision block 74 that the flow detached at block 70 was not the only flow

enqueued in the scheduling queue 42, then the procedure of FIG. 6 loops back directly to block 60 from decision block 74, i.e., without placing the empty indicator 50 associated with the scheduling queue 42 in a condition to indicate that the scheduling queue 42 is empty. (It is to be understood that in the event that more than one frame is enqueued in the flow queue corresponding to the flow detached at block 70, then the flow will be reattached to the scheduling queue 42 in due course according to conventional practice. Of course, upon the reattachment of the flow to the scheduling queue 42, the empty indicator 50, which had been placed in a condition to indicate the scheduling queue 42 was empty, will once again be placed in a condition to indicate that the scheduling queue 42 is not empty.)

The procedure of FIG. 6 is advantageous as compared to the procedure of FIG. 5, in that, with the procedure of FIG. 5, a scheduling queue may be emptied by detachment of the last flow therefrom, without the corresponding empty indicator being set to indicate that the scheduling queue is empty. By contrast, with the inclusion of blocks 72 and 74 in FIG. 6, detachment of the last flow from a scheduling queue is detected and the placement of the corresponding empty indicator in a condition to indicate that the scheduling queue is empty occurs immediately, thereby substantially eliminating indications that a scheduling queue is not empty when in fact it is empty. (Block 64 remains desirable in the procedure of FIG. 6, however, to deal with rare "race" conditions in which an empty indicator does not indicate a scheduling queue is empty in time to avoid a false indication that the scheduling queue is not empty.) The processes of FIGS. 5

and 6 may be implemented in hardware, software or a combination thereof.

In at least one embodiment of the invention, the processes of FIGS. 5 and 6 are implemented in hardware employing a suitable combination of conventional logic circuitry such as adders, comparators, selectors, etc. Such hardware, for example, may be located within the scheduler 49 (FIG. 4). A person of ordinary skill in the art may develop logic circuitry capable of performing the inventive processes described with reference to FIGS. 5 and 6. In a software embodiment of the invention, the processes of FIGS. 5 and 6 may comprise one or more computer program products. Each inventive computer program product may be carried by a medium readable by a computer (e.g., a carrier wave signal, a floppy disk, a hard drive, a random access memory, etc.).

The empty indicator arrangement of the present invention provides an efficient and cost effective way of identifying empty scheduling queues before they are searched. Consequently, operating cycles of a scheduler employing the inventive empty indicators are less likely to be wasted in searching a scheduling queue that is empty.

The foregoing description discloses only exemplary embodiments of the invention; modifications of the above disclosed apparatus and methods which fall within the scope of the invention will be readily apparent to those of ordinary skill in the art. For example, in the embodiments described above, scheduling queues are maintained in a separate scheduler chip associated with a network processor. However, it is also contemplated that scheduling queues may be maintained in a scheduler circuit that is implemented as

part of a data flow chip or as part of a processor chip in a network processor.

Accordingly, while the present invention has been disclosed in connection with exemplary embodiments thereof, it should be understood that other embodiments may fall within the spirit and scope of the invention as defined by the following claims.